Music analysis and deep learning

Brian McFee

2016.04.05

NYU | Center for Data Science

MARL

Plan for today

1. What do we want to do with music?
2. How does music differ from other domains?
3. Overview of traditional analysis pipelines
4. Deepification
5. Example applications
6. Tips, tricks, and resources

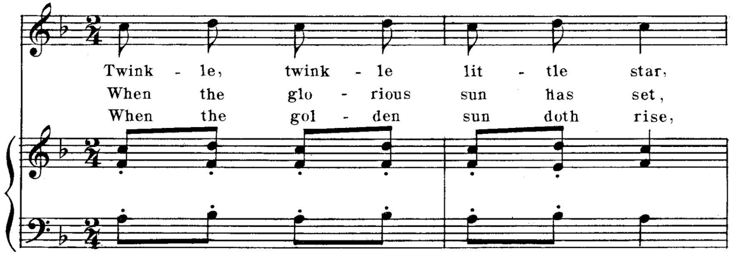
What can we do with (or to) music?

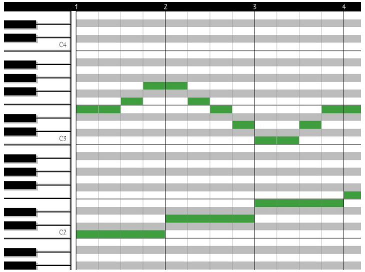
Why analyze music with computers?

- Recommender systems (Spotify, Pandora, etc.)
- Musicology / discography / archival applications
- Educational tools
- Visualization / interactivity
- Creative applications

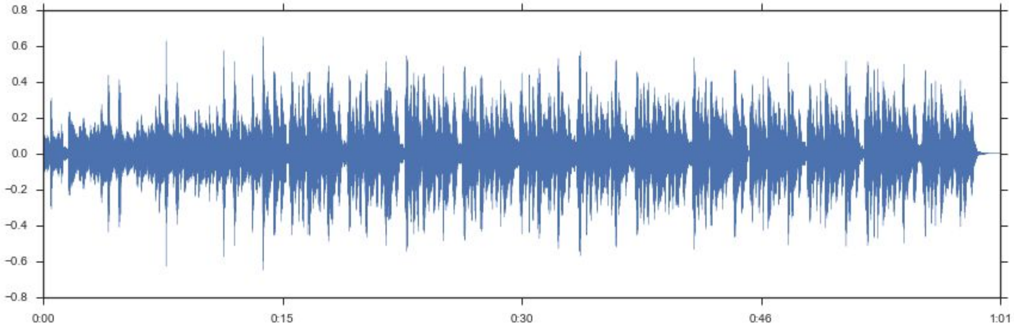
What do we mean by music?

- Score / symbolic representation





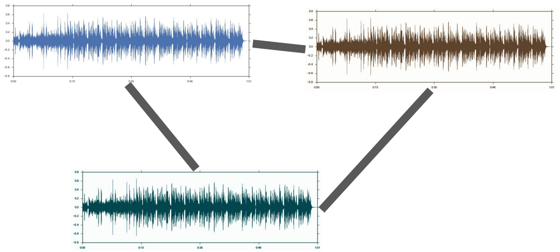
- **Audio**



Applications of music analysis: corpus level

- Recommendation / similarity

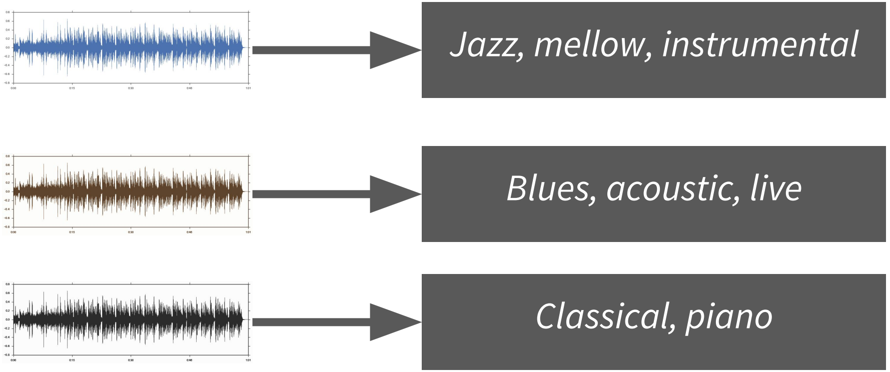
- Fingerprinting / duplicate detection



- Cover / composition identification

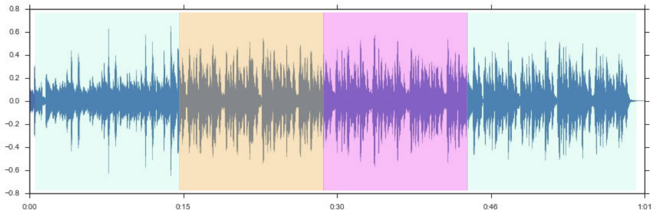
Applications of music analysis: track level

- Auto-tagging / search and retrieval



Applications of music analysis: time-varying

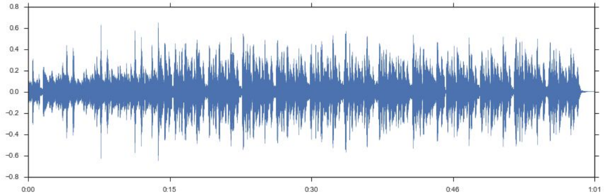
- Instrument detection
- Chord recognition
- Beat / meter tracking
- Structural segmentation
- Transcription



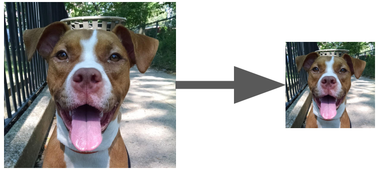
What’s special about music?

Audio vs. Images

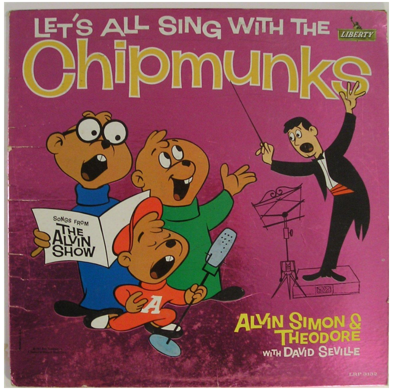
- 1D Input
  - amplitude(time)



- **Not** scale-invariant
  - downsampling/pooling does not preserve structures like in images



- Models should support variable-length input



- Interactions can be local **or long-range**

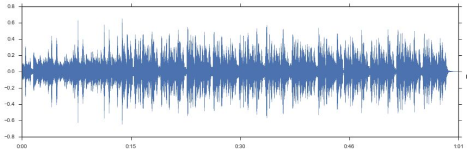
Music vs. Speech

- Sources overlap in time and frequency
- “Ground truth” usually doesn’t exist
- Input is more varied
- Repetition is important

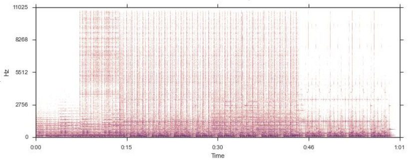
Traditional analysis pipelines

A standard pipeline, ca. 2011

Audio



Spectrogram



Perceptual model

Basis projection
Log amplitude

Statistical model

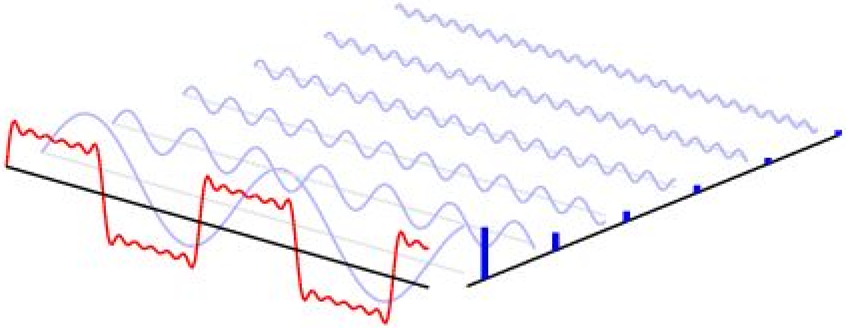
GMM, K-means,
HMM, SVM, LR, ...

Predictions

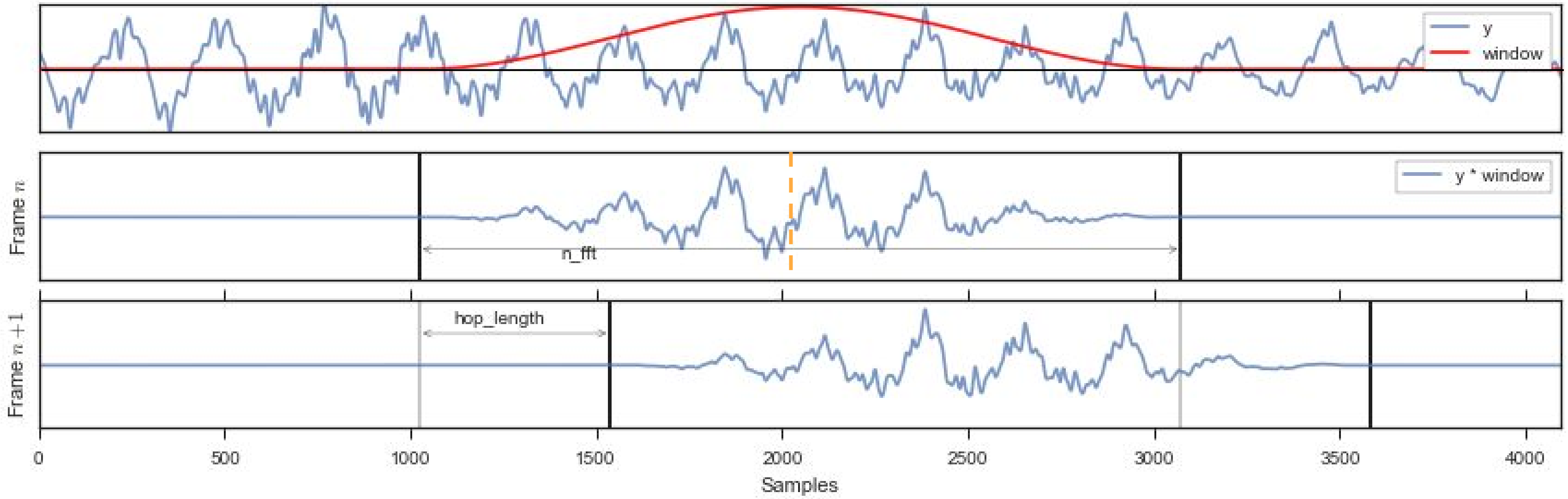
(e.g., instruments, tags, beats, ...)

Spectrograms

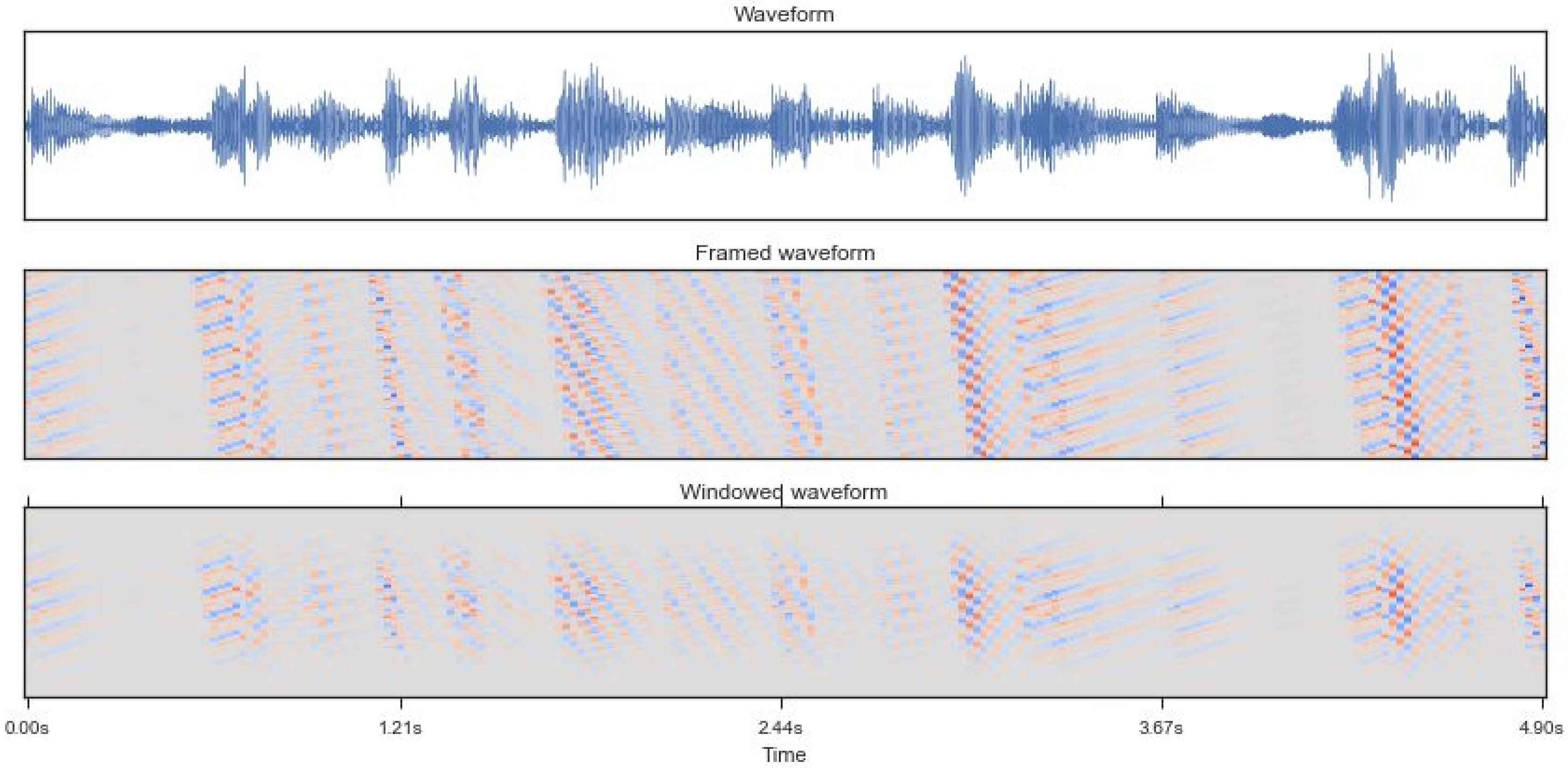
- Break sound apart into different frequencies
    - **Fourier analysis**
- Measure how energy changes over time
    - **Short-time Fourier transform**
- Break audio into small, overlapping **frames**



http://pgfplots.net/tikz/examples/fourier-transform/



frame[t] → frame[t] * window[t]



Note: neighboring frames differ mostly by shifts

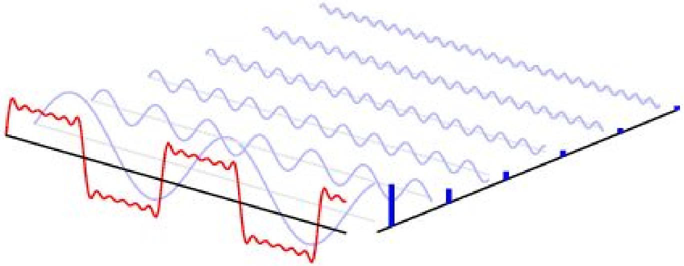
Fourier transform

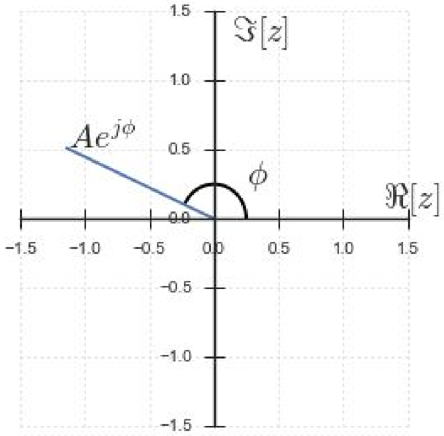
- **Complex**-valued coefficient for each frequency

  $$X(f) = Ae^{j\varphi}$$

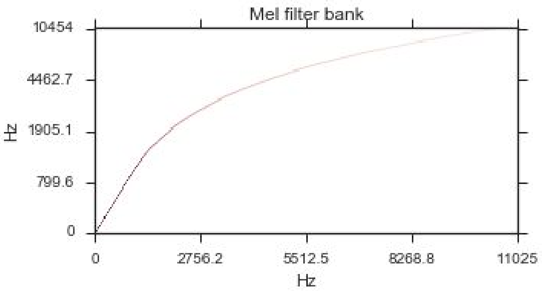
  - A = Magnitude (energy)
  - φ = Phase (shift)

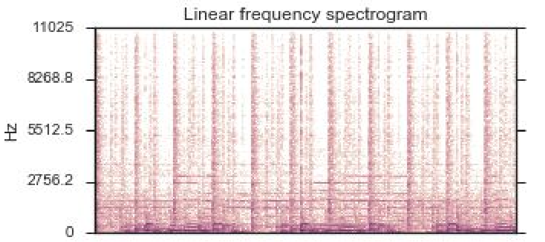
- Usually we discard phase and analyze only the magnitude
  - Raw phase is non-linear, unstable wrt frame alignment
  - Complex math is hard :(
  - Energy carries a lot of information anyway

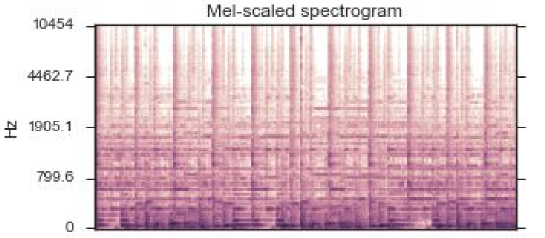




Perceptual model: frequency



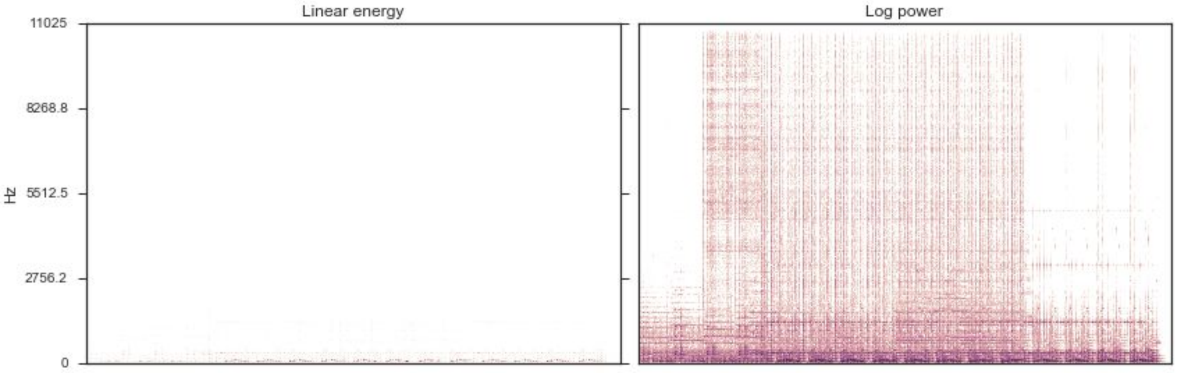




- FFT uses **linearly**-spaced frequencies
- Mel frequency scale is **log**-spaced above a threshold frequency
  - Tuned by psychoacoustic experiments
- Often used as **dimensionality reduction**
  - Eg, 1025 FFT bins -> 128 Mel bins
- Decent representation for timbre, not so great for pitch

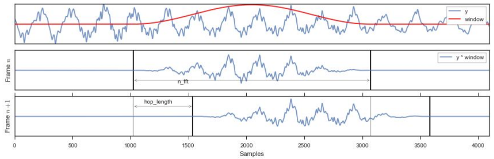
Perceptual model: amplitude

- Perception of “loudness” is approximately logarithmic, not linear
- Convert $|A|$ to $\log |A|^2$
- Linearizes energy ratio comparisons

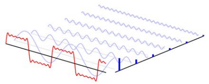


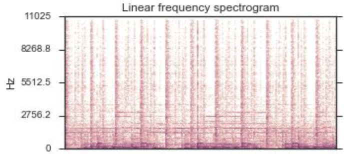
What did all of this do?

1. Framing = local feature extraction + downsampling

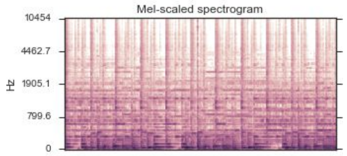


2. FFT = linear change of basis
    a. Magnitude spectrum = non-linearity





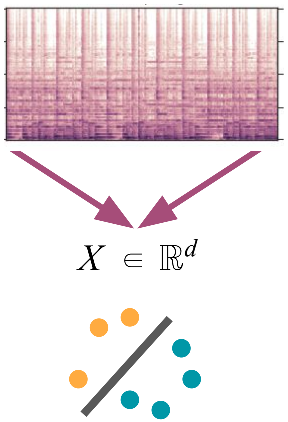
3. Mel-scaling = dimensionality reduction

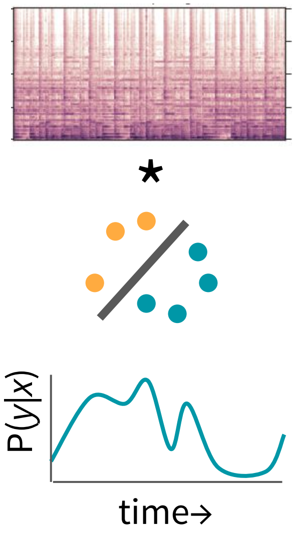


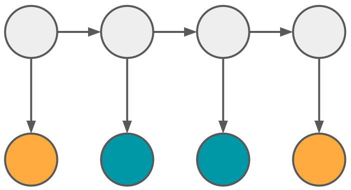
4. Log amplitude = non-linearity

- **End result**: time-series of feature vectors

	Global output	Time-varying output (local)	Time-varying output (long-range)
Example	*Tag prediction*	*Instrument activations*	*Chord recognition*
Method	• Summarize features over time • Model the summary vector	• Predict on local feature windows • I.e., convolution with classifier	• Hidden Markov Model or DP







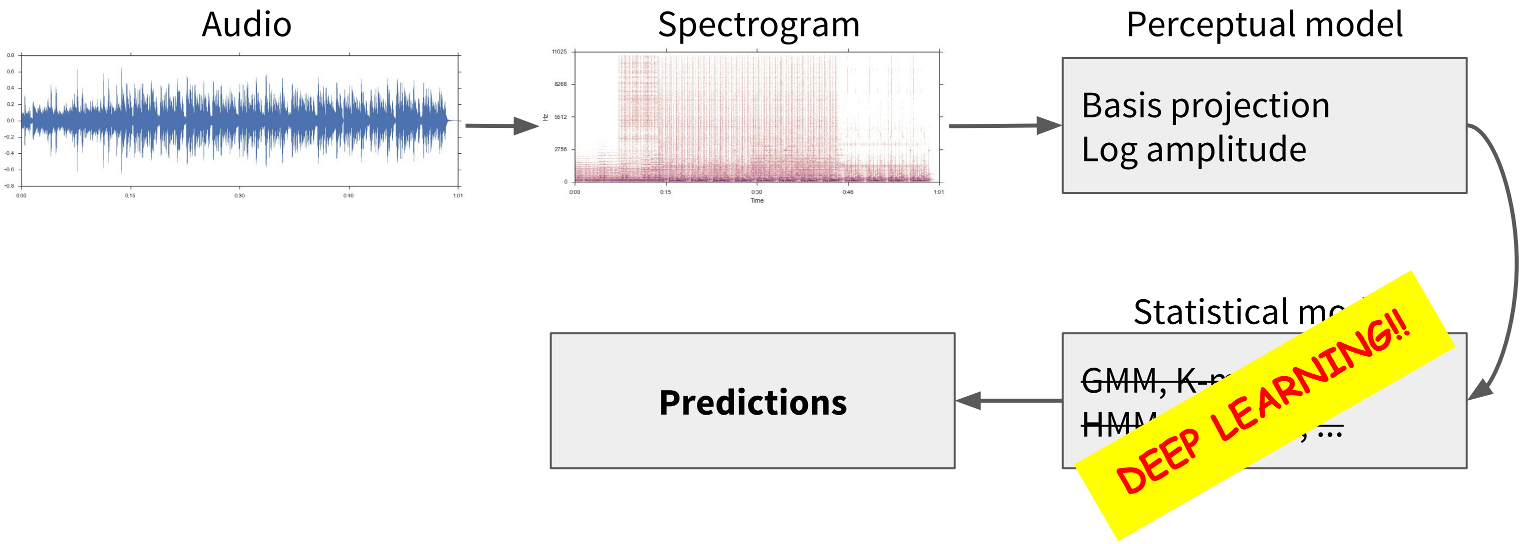
Prediction time

Deepification

Going deep: option 1

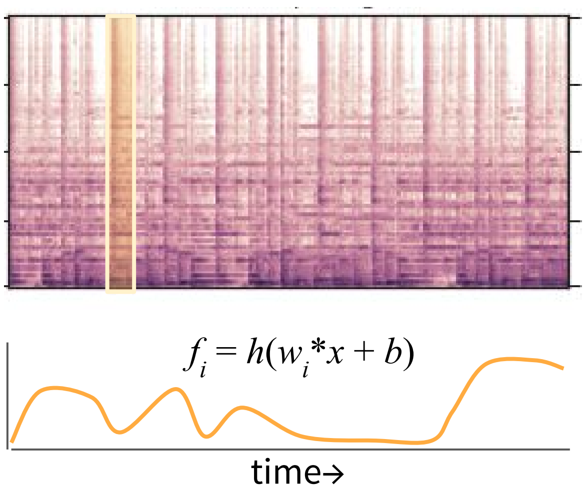
- Stick a deep network on top of the existing feature stack

- Add time-convolution or recurrence to exploit temporal dependencies



Common architectures: convnets

- Whitening (or batch-norm) after log-scaling
- Full-height filters = time-convolution
- For global prediction tasks, temporal pooling is fine
- For time-varying tasks, time-pooling reduces output resolution
  - Dense layers should be avoided
- After first conv layer, looks like any other convnet



Some issues with that approach...

- It can only exploit time-shift invariance.
  - What about pitch-invariance?
- Mel scaling is lossy dimensionality reduction.
  - Can we do better? Do we need it at all?
- Do we even need FFTs? Why not just learn from raw audio?
- That said, this approach is often effective!

Pitch invariant features

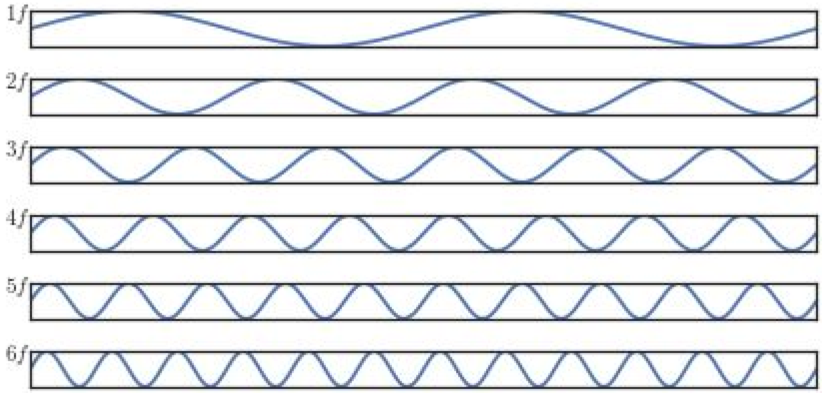
- We want

  Constant vertical shift = constant change in pitch

- This requires **log-spaced** frequencies
  - E.g., piano equal temperament scale:

    $$f_{i+1} = 2^{1/12} f_i$$

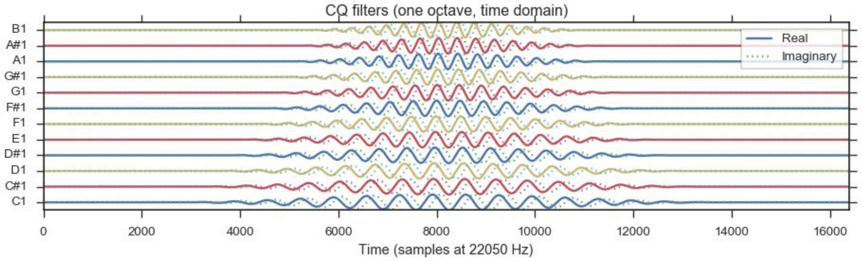
- Fourier basis does not do this



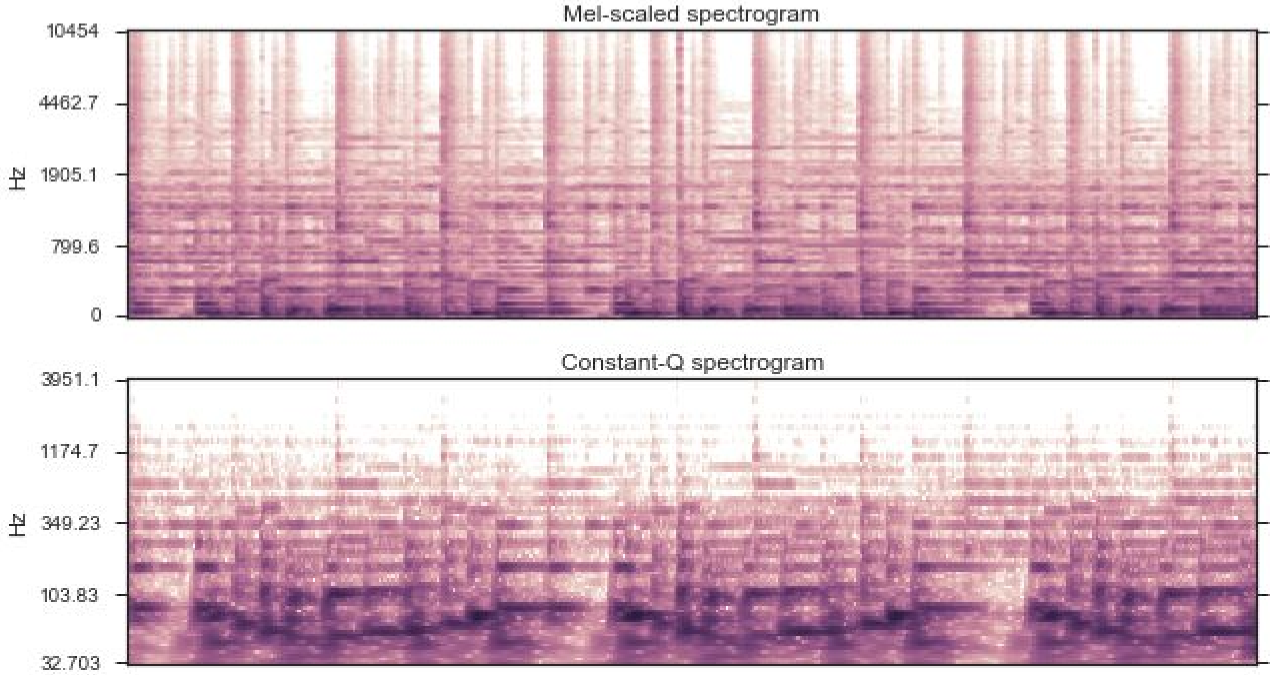
- Mel only does this at the high end
  - Poor bass resolution

Constant-Q Transform

- **Log-scaled** frequency representation



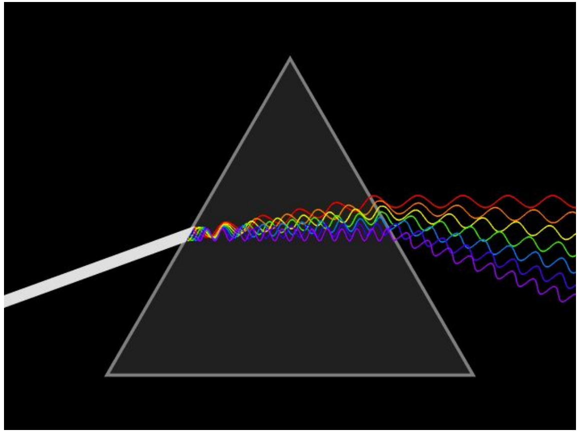
- Slightly lossy (compared to FFT)
  - but still invertible
- Works by using different-length windows for each frequency
- Good representation for both pitch and timbre


Mel-scaled spectrogram
10454
4462.7
1905.1
799.6
0
Hz
Constant-Q spectrogram
3951.1
1174.7
349.23
103.83
32.703
Hz

Mel vs CQT

What about learning from raw audio?

- People have tried
  - [Dieleman and Schrauwen, 2014]
  - [Tsainath et al., 2015] (speech)
- It's possible, but not always worth it
  - May require many more filters to match performance
- FFT and CQT have nice properties that we understand
- Nobody complains about frequency decompositions for images...

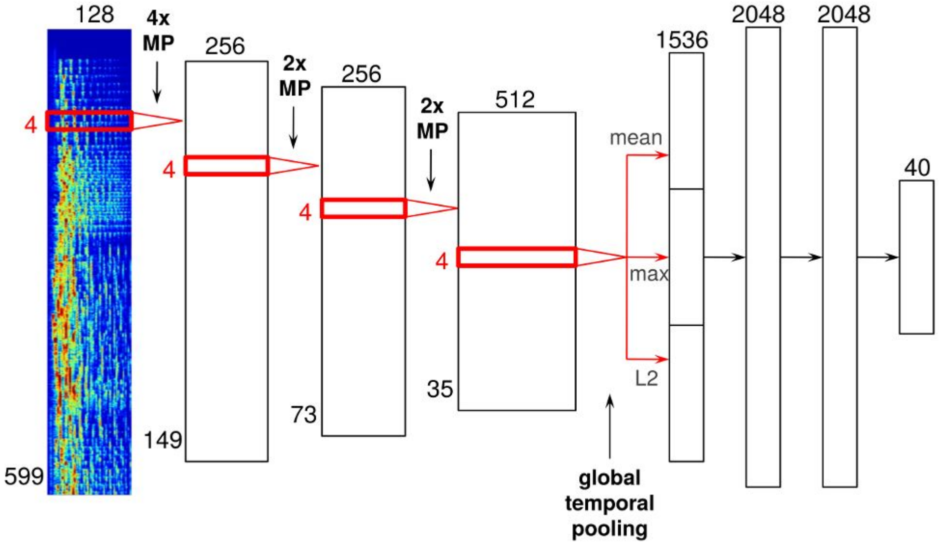


Example applications

[van den Oord, Dieleman, Schrauwen 2013]

Example 1: Similarity / recommendation

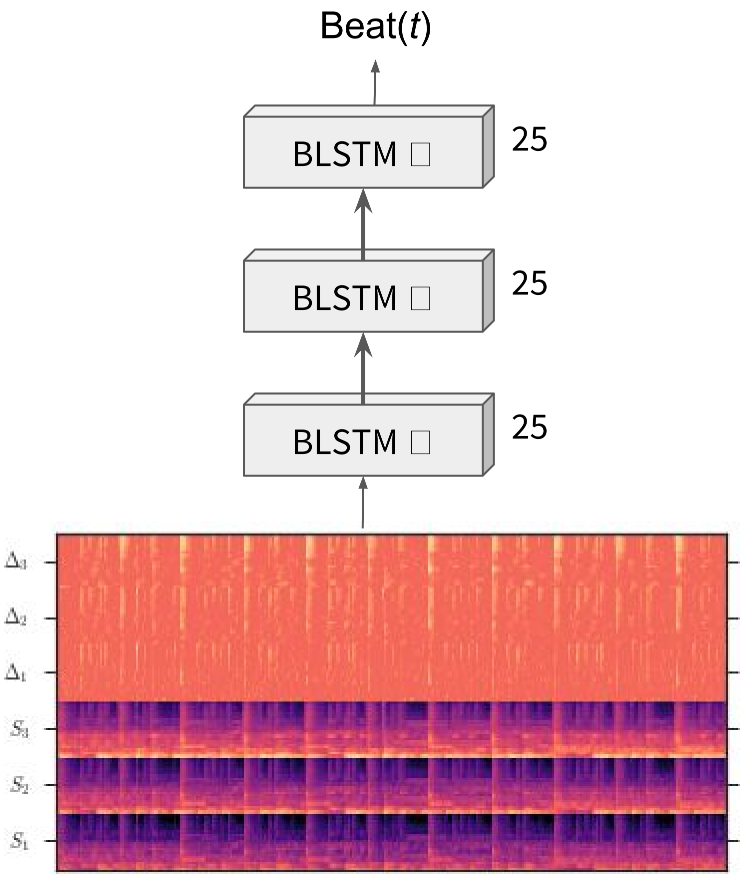
- Input:
  - 30 sec of log-Mel spectra (128x600)
- Output:
  - Vector in $\mathbb{R}^{40}$
  - Collaborative filter embedding
- Application:
  - Track-level similarity / recommendation
- Dataset:
  - MSD, private Spotify data



http://benanne.github.io/2014/08/05/spotify-cnns.html

Example 2: Beat tracking

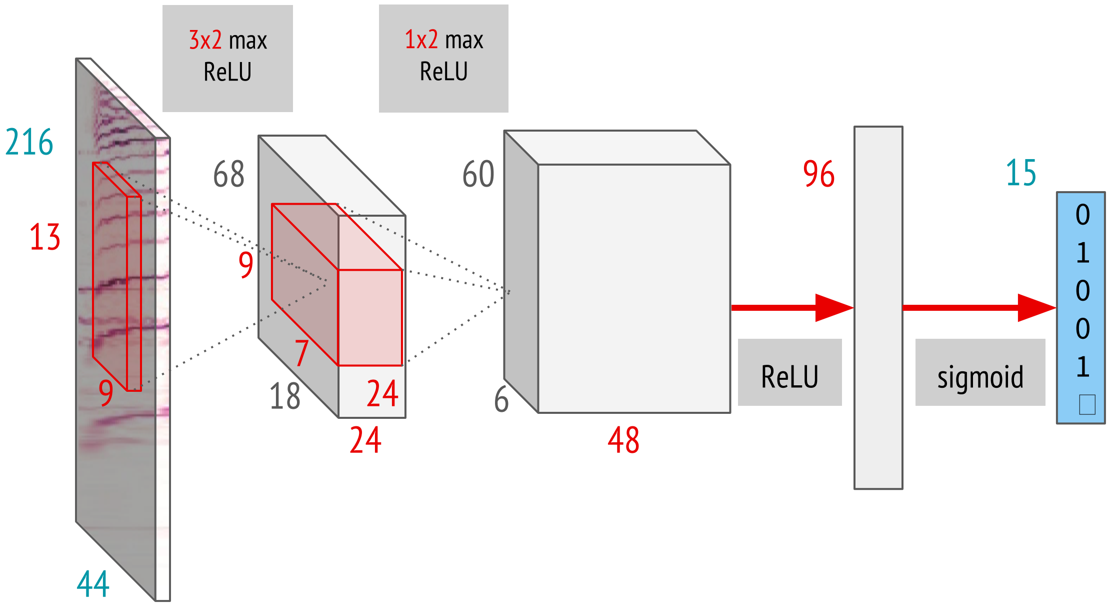
- Input:
  - Multi-resolution log Mel-spectra (20 bins * 3 resolution)
  - Thresholded deviation from local median (rising energy detector)
  - 120xT input data
- Output:
  - Time-series beat/not-beat
  - BLSTM + peak-picking heuristic
- Dataset:
  - “Ballroom set” (88 tracks)
  - MIREX2006 (26 + 6 tracks)
  - JPB (6 tracks)



[M., Humphrey, Bello 2015]

Example 3: Instrument identification

- Input:
  - 1sec CQT clip (216x44)
  - 36 bins per octave
  - 6 octaves (C2-C8)
  - log-scaled
- Output:
  - 15 binary instrument labels
- Dataset:
  - MedleyDB
  - + lots of data augmentation



Tips, tricks, resources

Software

- Librosa
  - Audio feature extraction in python
- Mir_eval
  - Standard metrics for music tasks
- Muda
  - Musical data augmentation
- JAMS
  - JSON annotated music specification

Data sets

- Million Song Dataset
  - Pre-computed features
  - Tags*, Lyrics*, Collaborative filter, Cover songs, Similar songs, ...
- MedleyDB
  - Separated sources (stems) **with audio**
  - Melody and instrument annotations
- SALAMI
  - Structure annotations, multiple annotators
- ISOPhonics
  - Chords, keys, beats, structure for Beatles + a few othres

General tips

- Understand where your data comes from!
- Think about what properties your model should exploit in the signal
- Always do error analysis:
  - when something doesn't work, find out why!
- People have thought a lot about DSP -- use that knowledge!

Questions?

brian.mcfee@nyu.edu